CIS 5500 – Milestone 3
Funding-Aware Market Maker

Gaurav Malhotra Albert Opher Ishaan Shah Madhav Sharma

Part 1: SQL Queries

Below we present ten SQL queries that run on our PostgreSQL database. Queries 1–4 are our designated complex queries for the purposes of Milestone 3.

Query 1 (Complex): Cumulative return around each funding event in [-60, +180] minutes

```
WITH funding_events AS (
    SELECT
        symbol,
        ts AS event_ts
    FROM funding
    WHERE symbol = $1                       -- symbol filter
      AND ts BETWEEN $2 AND $3              -- start_ts, end_ts
),
window_returns AS (
    SELECT
        f.symbol,
        f.event_ts,
        mr.ts,
        SUM(mr.r1m) OVER (
            PARTITION BY f.symbol, f.event_ts
            ORDER BY mr.ts
            ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
        ) AS cum_return
    FROM funding_events f
    JOIN minute_returns mr
      ON mr.symbol = f.symbol
     AND mr.ts BETWEEN f.event_ts - INTERVAL '60 minutes'
                   AND f.event_ts + INTERVAL '180 minutes'
)
SELECT
    symbol,
```

```
    event_ts,
    MIN(cum_return) AS min_car,
    MAX(cum_return) AS max_car
FROM window_returns
GROUP BY symbol, event_ts
ORDER BY event_ts;
```

Query 2 (Complex): Funding rate deciles by day vs 60-minute post-event return

```
WITH funding_with_decile AS (
    SELECT
        symbol,
        ts,
        rate,
        NTILE(10) OVER (
            PARTITION BY DATE(ts)
            ORDER BY rate
        ) AS rate_decile
    FROM funding
    WHERE ts BETWEEN $1 AND $2            -- date range
),
event_markouts AS (
    SELECT
        f.symbol,
        f.ts,
        f.rate_decile,
        SUM(mr.r1m) AS markout_60m
    FROM funding_with_decile f
    JOIN minute_returns mr
      ON mr.symbol = f.symbol
     AND mr.ts > f.ts
     AND mr.ts <= f.ts + INTERVAL '60 minutes'
    GROUP BY f.symbol, f.ts, f.rate_decile
)
SELECT
    rate_decile,
    AVG(markout_60m) AS avg_markout_60m,
    COUNT(*)         AS n_events
FROM event_markouts
GROUP BY rate_decile
ORDER BY rate_decile;
```

Query 3 (Complex): Extreme regime where funding > daily p90 and OI > rolling 14d p90

```
WITH daily_rate_stats AS (
    SELECT
        symbol,
        DATE(ts) AS d,
        PERCENTILE_CONT(0.9) WITHIN GROUP (ORDER BY ABS(rate)) AS p90_abs_rate
    FROM funding
    WHERE ts BETWEEN $1 AND $2            -- date range
    GROUP BY symbol, DATE(ts)
),
rolling_oi_stats AS (
    SELECT
        symbol,
        ts,
        PERCENTILE_CONT(0.9) WITHIN GROUP (ORDER BY oi) OVER (
            PARTITION BY symbol
            ORDER BY ts
            RANGE BETWEEN INTERVAL '14 days' PRECEDING AND CURRENT ROW
        ) AS p90_oi_14d
    FROM open_interest
    WHERE ts BETWEEN $1 - INTERVAL '14 days' AND $2
),
regime_events AS (
    SELECT
        f.symbol,
        f.ts
    FROM funding f
    JOIN daily_rate_stats dr
      ON dr.symbol = f.symbol
     AND dr.d = DATE(f.ts)
    JOIN rolling_oi_stats oi
      ON oi.symbol = f.symbol
     AND oi.ts = f.ts
    WHERE ABS(f.rate) > dr.p90_abs_rate
      AND oi.oi > oi.p90_oi_14d
),
event_markouts AS (
    SELECT
        r.symbol,
        r.ts,
        SUM(mr.r1m) AS markout_60m
    FROM regime_events r
    JOIN minute_returns mr
      ON mr.symbol = r.symbol
     AND mr.ts > r.ts
```

```
     AND mr.ts <= r.ts + INTERVAL '60 minutes'
    GROUP BY r.symbol, r.ts
)
SELECT
    symbol,
    AVG(markout_60m) AS avg_markout_60m,
    COUNT(*)         AS n_events
FROM event_markouts
GROUP BY symbol
HAVING COUNT(*) >= $3                    -- minimum # of regime events
ORDER BY avg_markout_60m DESC
LIMIT $4;                                -- top-K symbols
```

Query 4 (Complex): Symbols that never have negative 30m CAR in low-vol regimes

```
WITH funding_rv AS (
    SELECT
        f.symbol,
        f.ts,
        STDDEV_SAMP(mr.r1m) AS rv_1d
    FROM funding f
    JOIN minute_returns mr
      ON mr.symbol = f.symbol
     AND mr.ts BETWEEN f.ts - INTERVAL '1 day'
                   AND f.ts
    WHERE f.ts BETWEEN $1 AND $2         -- date range
    GROUP BY f.symbol, f.ts
),
median_rv AS (
    SELECT
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY rv_1d) AS med_rv
    FROM funding_rv
)
SELECT DISTINCT fr.symbol
FROM funding_rv fr, median_rv m
WHERE NOT EXISTS (
    SELECT 1
    FROM minute_returns mr
    WHERE mr.symbol = fr.symbol
      AND mr.ts > fr.ts
      AND mr.ts <= fr.ts + INTERVAL '30 minutes'
      AND mr.r1m < 0                 -- negative markout
      AND fr.rv_1d < m.med_rv        -- low-vol regime
```

```
)
ORDER BY fr.symbol;
```

Query 5: Average 60-minute markout by hour-of-day for funding events

```
WITH event_markouts AS (
    SELECT
        f.symbol,
        f.ts,
        SUM(mr.r1m) AS markout_60m
    FROM funding f
    JOIN minute_returns mr
      ON mr.symbol = f.symbol
     AND mr.ts > f.ts
     AND mr.ts <= f.ts + INTERVAL '60 minutes'
    WHERE f.ts BETWEEN $1 AND $2          -- date range
    GROUP BY f.symbol, f.ts
)
SELECT
    EXTRACT(HOUR FROM ts) AS funding_hour,
    AVG(markout_60m)      AS avg_markout_60m,
    COUNT(*)              AS n_events
FROM event_markouts
GROUP BY funding_hour
ORDER BY funding_hour;
```

Query 6: Markouts conditioned on short-term volatility regime

```
WITH event_vol AS (
    SELECT
        f.symbol,
        f.ts,
        STDDEV_SAMP(mr.r1m) AS rv_1h
    FROM funding f
    JOIN minute_returns mr
      ON mr.symbol = f.symbol
     AND mr.ts BETWEEN f.ts - INTERVAL '1 hour' AND f.ts
    WHERE f.ts BETWEEN $1 AND $2
    GROUP BY f.symbol, f.ts
),
event_vol_regimes AS (
    SELECT
        symbol,
        ts,
```

```
        rv_1h,
        NTILE(3) OVER (ORDER BY rv_1h) AS vol_regime   -- 1=low, 3=high
    FROM event_vol
),
event_markouts AS (
    SELECT
        f.symbol,
        f.ts,
        evr.vol_regime,
        SUM(mr.r1m) AS markout_60m
    FROM funding f
    JOIN event_vol_regimes evr
      ON evr.symbol = f.symbol AND evr.ts = f.ts
    JOIN minute_returns mr
      ON mr.symbol = f.symbol
     AND mr.ts > f.ts
     AND mr.ts <= f.ts + INTERVAL '60 minutes'
    WHERE f.ts BETWEEN $1 AND $2
    GROUP BY f.symbol, f.ts, evr.vol_regime
)
SELECT
    vol_regime,
    AVG(markout_60m) AS avg_markout_60m,
    COUNT(*)         AS n_events
FROM event_markouts
GROUP BY vol_regime
ORDER BY vol_regime;
```

Query 7: Overview of symbols: counts and basic liquidity stats

```
SELECT
    s.symbol,
    COUNT(DISTINCT k.open_time) AS n_klines,
    COUNT(DISTINCT f.ts)        AS n_funding_events,
    AVG(k.volume)               AS avg_kline_volume
FROM symbols s
LEFT JOIN klines k
  ON k.symbol = s.symbol
LEFT JOIN funding f
  ON f.symbol = s.symbol
WHERE k.open_time BETWEEN $1 AND $2       -- date range for stats
GROUP BY s.symbol
ORDER BY s.symbol;
```

Query 8: Rank symbols by average |funding rate|

```
SELECT
    symbol,
    AVG(ABS(rate)) AS avg_abs_rate,
    COUNT(*)       AS n_events
FROM funding
WHERE ts BETWEEN $1 AND $2                -- date range
GROUP BY symbol
HAVING COUNT(*) >= $3                     -- minimum # of funding prints
ORDER BY avg_abs_rate DESC
LIMIT $4;                                 -- top-K
```

Query 9: Average 30-minute realized volatility after funding

```
WITH event_rv AS (
    SELECT
        f.symbol,
        f.ts,
        STDDEV_SAMP(mr.r1m) AS rv_30m
    FROM funding f
    JOIN minute_returns mr
      ON mr.symbol = f.symbol
     AND mr.ts BETWEEN f.ts AND f.ts + INTERVAL '30 minutes'
    WHERE f.ts BETWEEN $1 AND $2          -- date range
    GROUP BY f.symbol, f.ts
)
SELECT
    symbol,
    AVG(rv_30m) AS avg_rv_30m,
    COUNT(*)    AS n_events
FROM event_rv
GROUP BY symbol
ORDER BY avg_rv_30m DESC;
```

Query 10: Count events where 30-minute CAR exceeds a threshold

```
WITH event_car AS (
    SELECT
        f.symbol,
        f.ts,
        SUM(mr.r1m) AS car_30m
    FROM funding f
    JOIN minute_returns mr
```

```
      ON mr.symbol = f.symbol
     AND mr.ts > f.ts
     AND mr.ts <= f.ts + INTERVAL '30 minutes'
    WHERE f.ts BETWEEN $1 AND $2          -- date range
    GROUP BY f.symbol, f.ts
)
SELECT
    symbol,
    COUNT(*) AS n_positive_moves
FROM event_car
WHERE car_30m > $3                        -- CAR threshold
GROUP BY symbol
ORDER BY n_positive_moves DESC;
```

Part 2: Query Descriptions

Below we provide a brief, natural-language description for each query.

Query 1:

This query computes the cumulative return path from 60 minutes before each funding timestamp to 180 minutes after, for a given symbol and date range. For each funding event, it reports the minimum and maximum cumulative return in that window, allowing us to detect drift or reversal patterns around funding.

Query 2:

This query assigns each funding print to a daily funding-rate decile using NTILE(10), then computes the 60-minute post-event return for each funding event. It aggregates by decile to show how average 60-minute drift varies across funding-rate buckets (e.g., whether the highest deciles exhibit stronger directionality).

Query 3:

This query identifies "stress" regimes where a funding event's absolute rate is above its symbol/day 90th percentile and open interest is above its rolling 14-day 90th percentile. It then computes 60-minute post-event drift for these regime events and ranks symbols by their average drift, filtered to symbols with at least a minimum number of such events.

Query 4:

This query first computes 1-day realized volatility prior to each funding event and the global median of these volatilities. It then returns symbols for which there does not exist any funding event where volatility is below the median and the 30-minute post-event return is negative, effectively implementing a universal NOT EXISTS condition over low-volatility environments.

Query 5:

This query computes the 60-minute markout after each funding event and groups results by the hour-of-day of the funding timestamp. It returns the average 60-minute drift and the number of events per hour, allowing us to detect intraday patterns in how prices respond to funding.

Query 6:

This query computes 1-hour realized volatility before each funding event and uses NTILE(3) to divide events into low, medium, and high volatility regimes. It then computes average 60-minute drift per regime, letting us compare how post-funding behavior changes with short-term volatility.

Query 7:

This query reports, for each symbol, the number of klines, the number of funding events, and the average kline volume over a specified date range. It provides a quick overview of dataset coverage and liquidity across symbols.

Query 8:

This query computes the average absolute funding rate per symbol over a given date range, restricted to symbols with at least a minimum number of funding prints. It ranks symbols by this metric, highlighting assets with persistently high funding pressure.

Query 9:

This query computes the 30-minute realized volatility (standard deviation of one-minute returns) after each funding event and then averages this volatility per symbol. It allows us to see which symbols tend to experience more volatile post-funding periods.

Query 10:

This query computes the 30-minute cumulative return after each funding event and counts, for each symbol, how often this CAR exceeds a chosen positive threshold. It is useful for identifying symbols where funding events are more frequently followed by strong positive moves.

Part 3: Database Credentials and Repository Link

AWS RDS Connection Information

- **Host:** `cis550-project-instance.c5m282o04n2q.us-east-1.rds.amazonaws.com`
- **Database Name:** `cis550_project`
- **Port:** `5432`
- **Username:** `postgres`
- **Password:** `m2wurbpn`

Private GitHub Repository

- **Repository URL:** `https://github.com/Madhav-Sharma-07/cis5500_project`
- **Access Granted To:** `dhruv4321github`